

# The influence of parsimony and randomness on complexity growth in Tierra

Russell K. Standish

School Mathematics, UNSW, 2052 Australia

<http://parallel.hpc.unsw.edu.au/rks>, [r.standish@unsw.edu.au](mailto:r.standish@unsw.edu.au)

## Abstract

The issue of how to create open-ended evolution in an artificial system is one of the open problems in artificial life. This paper examines two of the factors that have some bearing on this issue, using the Tierra artificial life system.

*Parsimony pressure* is a tendency to penalise more complex organisms by the extra cost needed to reproduce longer genotypes, encouraging simplification to happen. In Tierra, parsimony is controlled by the `SlicePow` parameter. When full parsimony is selected, evolution optimises the ancestral organism to produce extremely simple organisms. With parsimony completely relaxed, organisms grow larger, but not more complex. They fill up with “junk”. This paper looks at scanning a range of `SlicePow` from 0.9 to 1 to see if there is an optimal value for generating complexity.

Tierra (along with most ALife systems) use pseudo random number generators. Algorithms can never create information, only destroy it. So the total complexity of the Tierra system is bounded by the initial complexity, implying that the individual organism complexity is bounded. Biological systems, however, have plenty of sources of randomness, ultimately dependent on quantum randomness, so do not have this complexity limit. Sources of real random numbers exist for computers called *entropy gatherers* — this paper reports on the effect of changing Tierra’s pseudo random number generator for an entropy gatherer.

## Introduction

The issue of how to create open-ended evolution in an artificial system is one of the open problems in artificial life. This paper examines two of the factors that have some bearing on this issue, using the Tierra artificial life system (Ray, 1991)<sup>1</sup>. Tierra is well known artificial system, and well described in the literature, so only brief details will be given here. The digital organisms in Tierra consist of self-replicating codes written in a specially designed machine language. The Tierra environment is a virtual operating system executing the organism’s code in a time shared manner. Evolution proceeds through deliberately introduced mutations, copying errors and instruction flaws. Organisms compete for CPU time and memory space (called *soup*).

*Parsimony pressure* is a tendency to penalise more complex organisms by the extra cost needed to reproduce longer genotypes, encouraging simplification to happen. In Ray’s earliest experiments with Tierra, CPU time was allocated evenly between organisms, favouring organisms with the shortest genomes. The time sharing system was changed so that CPU time was allocated proportional to  $\ell^{\text{SlicePow}}$ . When `SlicePow`=0, we have the original maximal parsimony pressure. When `SlicePow`=1, parsimony pressure is removed entirely. In this case, organism length rapidly increases, until the soup consists of one organism whose length is greater than half of Tierra’s memory. At this point, it can no longer reproduce, and the soup dies (simulation stops).

But do organisms get more complex? For this purpose, we define complexity to be the *algorithmic information* (Li and Vitányi, 1997) of the organism. Adami (1998) introduced this measure in an artificial life setting, and I (Standish, 1999) developed a technique for measuring this in the Tierra setting. In (Standish, 2003), I report the first detailed study of a Tierra run. Whilst organisms get longer, their complexity shows no sign of increase at all. Their length comes from adding “junk” into their genomes.

Obviously neither extremes of `SlicePow` leads to complexity growth, but what if we tuned the parsimony pressure to modest values? In previous experiments, I knew that `SlicePow`< 0.9 led to shorter genomes, not longer, so in this paper, I scan a range of `SlicePow` from 0.9 to 1 to see if there is an optimal value for generating complexity.

Tierra (along with most ALife systems) use pseudo random number generators. Pseudo random number generators are short algorithms satisfying certain statistical tests for uniformity and independence. However, being the product of an algorithm, the output of a pseudo random number generator is not random by definition (Li and Vitányi, 1997). Algorithms can never create information, only destroy it. The complexity of any sequence of numbers is closely related to the length of the shortest algorithm that produces it, so the total complexity of the Tierra system with pseudo random number generators is bounded by its initial com-

<sup>1</sup>Available from <http://www.his.atr.jp/~ray/tierra/>

plexity, implying that the individual organism complexity is bounded. Biological systems, however, have plenty of sources of randomness, ultimately dependent on quantum randomness, so do not have this complexity limit.

The only way an algorithm can generate unbounded complexity is if it is coupled to a source of real randomness — a *random oracle*. Random oracles feature in Douglas Adams’s description of the *infinite improbability drive*:

*a Bambleweeny 57 Sub-meson Brain coupled to an atomic vector plotter suspended in a strong Brownian Motion producer (say a nice hot cup of tea)* (Adams, 1979, Chapt. 10).

It turns out to be simple enough to create random oracles: Geiger counters attached to radioactive sources (Gude, 1985)<sup>2</sup> and Lava lamps<sup>3</sup> are available through the internet to provide sources of genuine randomness, however these sources are limited to about 30 bytes per second.

Computers themselves have many different sources of random data available. They often interact with the external environment (eg users using keyboards, mice etc), and there is a small amount of randomness in timings in the hard disk (Jakobsson et al., 1998). Programs that harvest these sources of physical randomness are called *entropy gatherers*. Since unpredictability is important for cryptographic applications, practical true random number generation has experienced a lot of development in recent years. For example, the Linux operating system includes an entropy gatherer in its kernel, available as a character device at `/dev/random`.

Unfortunately, entropy gatherers, like the internet available random oracles, tend to be slow producers of randomness. HAVEGE (Seznec and Sendrier, 2003)<sup>4</sup> exploits many different sources of entropy with a modern computing system using hand crafted assembly language routines to increase the rate of entropy production by 3-4 orders of magnitude over the techniques available in `/dev/random`. This random stream is then used to seed a lookup table accessed by a pseudo random number generator to produce random numbers at similar rates to traditional pseudo random number generators. The entropy of the resulting sequence is less than a truly random sequence, but considerably higher than a pseudo random generator.

In this paper, we replace the pseudo random number generator in Tierra by calls to the HAVEGE generator, and compare what difference this makes to growth of complexity.

## Measurement of Complexity in Tierra

The most general definition of complexity of an object involves two levels of description, a *micro*-description which is its implementation, and a *macro*-description which is the

abstract *meaning* of an object. More than one microdescription can correspond to the same macrodescription. If  $\omega(\ell, x)$  is the number of microdescriptions of length  $\ell$  corresponding to macrodescription  $x$ , then the complexity of  $x$  is given by (Standish, 2001):

$$C(x) = \lim_{\ell \rightarrow \infty} \ell \log N - \log \omega(\ell, x). \quad (1)$$

where  $N$  is the size of the alphabet used for the microdescription. Eq (1) converges extremely rapidly for  $\ell > C(x)/\log N$ . The base of the logarithm determines what units you are measuring complexity in — if it is base 2, then the units are bits. For convenience, in this paper we will use base 32, corresponding to the alphabet size of the Tierra instruction set. Complexity is then measured in *instructions*. In order to measure the complexity of an organism in Tierra, we simply need to count up the number  $\omega(\ell, x)$  of computer programs of length  $\ell$  that are equivalent to a given digital organism  $x$ .

Not so simple! The first problem is how to determine if two computer programs are equivalent. The technique we use (Standish, 2003), is to record the results of a tournament where an organism is pitted pairwise against all genotypes recorded from a given Tierra run. Since this includes the context that these organisms experienced, any difference between two organism is expected to show up as a difference in the results of the two tournaments.

The second problem is that the number of programs of length  $\ell$  is  $32^\ell$ , a computationally infeasible number. In (Standish, 2003), I show that an alternative measure  $C_{ss}$  is a good first order estimate of the organismal complexity:

$$C_{ss} = \ell - \sum_{i=1} \log_{32} n_i \quad (2)$$

where  $n_i$  is the number of mutations at site  $i$  on the genome that lead to differing phenotypes.

This quantity is now very tractable, with a complete analysis of a  $10^{10}$  timestep Tierra run taking only a few hours on ac3’s Linux cluster — comparable to the time taken to perform the original Tierra run.<sup>5</sup>

## Results

Tierra was run with `SlicePow=0.9, 0.91...1.0` for  $10^{10}$  timesteps (instructions executed), with a soup size of 131072, both with the original random number generator, and with HAVEGE. When `SlicePow=1`, the runs terminated early due to the soup dying. Figure 1 shows the results for `SlicePow=0.96` using the Havege generator, which produced the maximum complexity of any run.

Figure 2 shows the maximum value of  $C_{ss}$  recorded for each run, as a function of parsimony pressure. It is showing fairly clearly that a `SlicePow` value between 0.95–0.96 is

<sup>2</sup><http://www.fourmilab.ch/hotbits/>

<sup>3</sup><http://www.lavarnd.org/>

<sup>4</sup><http://www.irisa.fr/caps/projects/hipsor/HAVEGE.html>

<sup>5</sup>The analysis code is available from <http://parallel.hpc.unsw.edu.au/getaegisdist.cgi/getdeltas/eco-tierra.3>

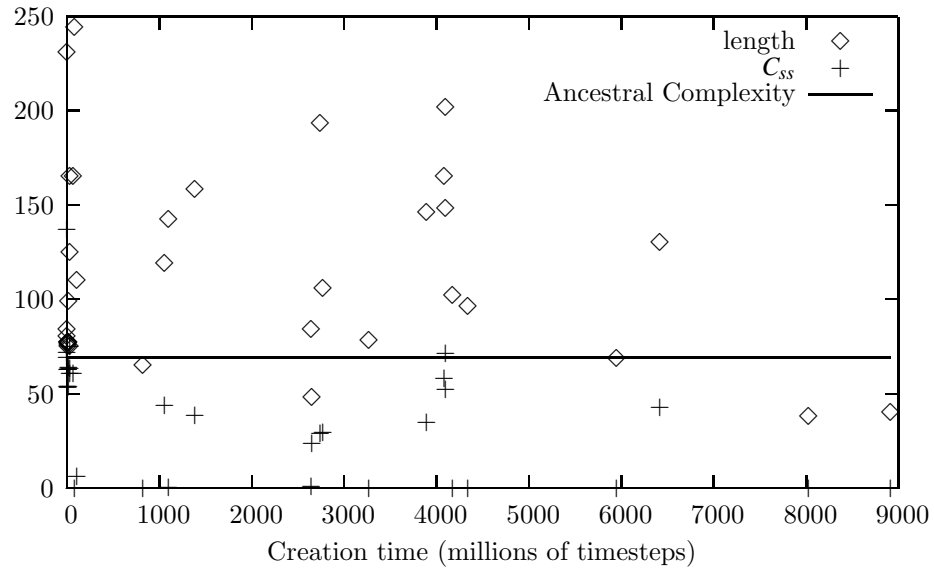


Figure 1: Plot of length and  $C_{ss}$  for the 36 unique phenotypes created during the run with  $\text{SlicePow}=0.96$ , and the HAVEGE entropy source. Three organisms appear with complexity greater than the ancestral organism 0080aaa within the first fifty million timesteps, including one with a complexity nearly twice that of the ancestor, and one appears at  $4.1 \times 10^9$  after which only simple organisms originate.

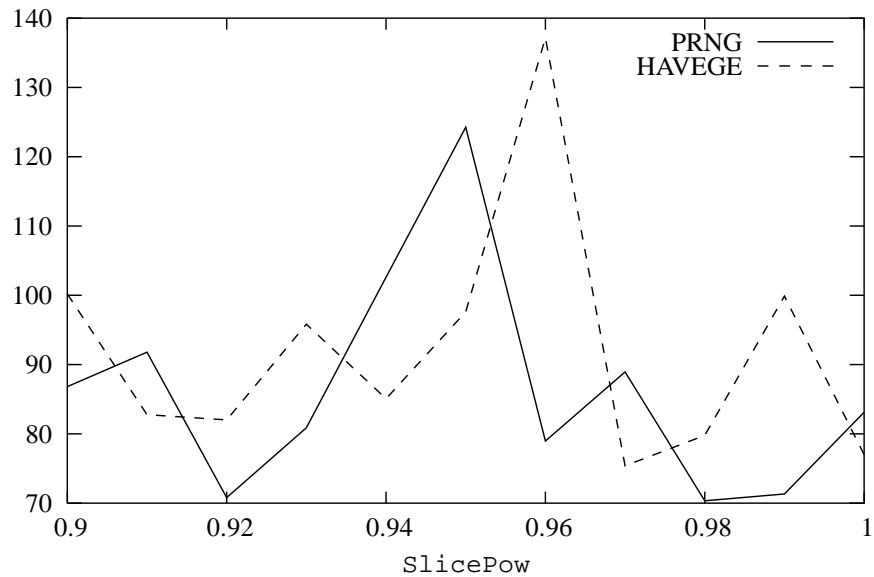


Figure 2: Plot of the maximum  $C_{ss}$  recorded as a function of parsimony pressure. PRNG = pseudo random number generator, and HAVEGE is the entropy harvester mentioned in the text.

needed to generate additional complexity. It is also suggestive that the entropy gatherer generates additional complexity over the pseudo random number generator, however this effect is unlikely to be statistically significant in this data set.

A different view of the data can be seen in figure 3, where the origination of the organism of maximal complexity is plotted. The interesting thing here is that the pseudo random number generator took a lot longer to find its maximally complex organism than the entropy gatherer algorithm.

## Discussion

The results reported here are of a small scale pilot study to study the effect of parsimony and of random oracles. Obviously much work needs to be done to tighten up the methodology of the experiment, and to perform analysis of statistical significance on the results. Clearly, this experiment does not show evidence of open-ended complexity growth either. Nevertheless, these interim results look encouraging.

Since the Tierra simulations are run in parallel on a Linux cluster, and it doesn't matter if one uses the same sequence of random numbers in each simulation, it should be possible to combine the entropy harvested from all CPUs, as well as network latency on the interconnect to substantially increase the entropy production of HAVEGE. This will require substantial recoding of HAVEGE to exploit this.

In the history of Earth's biosphere, complexity of individuals remain largely static for the first 2 billion years of life. It was only with originations of Eukaryotes circa 2Gya and of multicellular life circa 600Mya that we have any appreciable jump in complexity over the previous 2 billion years of bacterial life. During Phanerozoic (540Mya–present), there is little unambiguous evidence of complexity growth of organisms (McShea, 1996). However, what is a very clear trend is growth in the complexity of ecosystems. Diversity of the Earth's biosphere appears to have grown exponentially since Cambrian times (Benton, 2001).

Looking at things another way, a multicellular animal can be considered as an ecosystem of eukaryotic cells (OK so the genetic code for most of the cells is identical — gut flora being the obvious exception), and each eukaryotic cell can be considered an ecosystem of bacterial cells (nucleus + organelles). If anything, the “parts” of the World's biosphere have gotten simpler — it's the network connecting the parts that shown the complexity growth. In the Tierra case, what we should be looking for is overall complexity of the ecosystem, not complexity of the individual digital organisms.

At present, we still don't have a good theory for how to measure the complexity of an ecosystem, knowing its foodweb. Diversity is a lower bound on complexity, but is a rather crude indicator of overall complexity. Bedau-Packard (Bedau et al., 1998) statistics use diversity as one of the key indicators of open-ended evolution. This is probably all that is ever likely to be available for the Earth's biosphere, but in artificial life systems we can look for better measures of

ecosystem complexity.

## Acknowledgment

I would like to thank the *Australian Centre for Advanced Computing and Communications* for a grant of time on their Linux cluster for performing this work.

## References

- Adami, C. (1998). *Introduction to Artificial Life*. Springer.
- Adams, D. (1979). *Hitch Hikers Guide to the Galaxy*. Pan Books, London.
- Bedau, M. A., Snyder, E., and Packard, N. H. (1998). A classification of long-term evolutionary dynamics. In Adami, C., Belew, R., Kitano, H., and Taylor, C., editors, *Artificial Life VI*, pages 228–237, Cambridge, Mass. MIT Press.
- Benton, M. J. (2001). Biodiversity on land and in the sea. *Geological Journal*, 36:211–230.
- Gude, M. (1985). Concept for a high-performance random number generator based on physical random phenomena. *Frequenz*, 39:187–190.
- Jakobsson, M., Shriver, E., Hillyer, E., and Juels, A. (1998). A practical secure physical random bit generator. In *Proceedings of the 5th ACM Conference on Computer and Communications Security*, pages 103–111, San Francisco.
- Li, M. and Vitányi, P. (1997). *An Introduction to Kolmogorov Complexity and its Applications*. Springer, New York, 2nd edition.
- McShea, D. W. (1996). Metazoan complexity and evolution: Is there a trend? *Evolution*, 50:477–492.
- Ray, T. (1991). An approach to the synthesis of life. In Langton, C. G., Taylor, C., Farmer, J. D., and Rasmussen, S., editors, *Artificial Life II*, page 371. Addison-Wesley, Reading, Mass.
- Seznec, A. and Sendrier, N. (2003). HAVEGE: A user-level software heuristic for generating empirically strong random numbers. *ACM Transactions on Modeling and Computer Simulation*, 13:334–346.
- Standish, R. K. (1999). Some techniques for the measurement of complexity in Tierra. In Floreano, D., Nicoud, J.-D., and Mondada, F., editors, *Advances in Artificial Life: 5th European Conference, ECAL 99*, volume 1674 of *Lecture Notes in Computer Science*, page 104, Berlin. Springer.
- Standish, R. K. (2001). On complexity and emergence. *Complexity International*, 9.

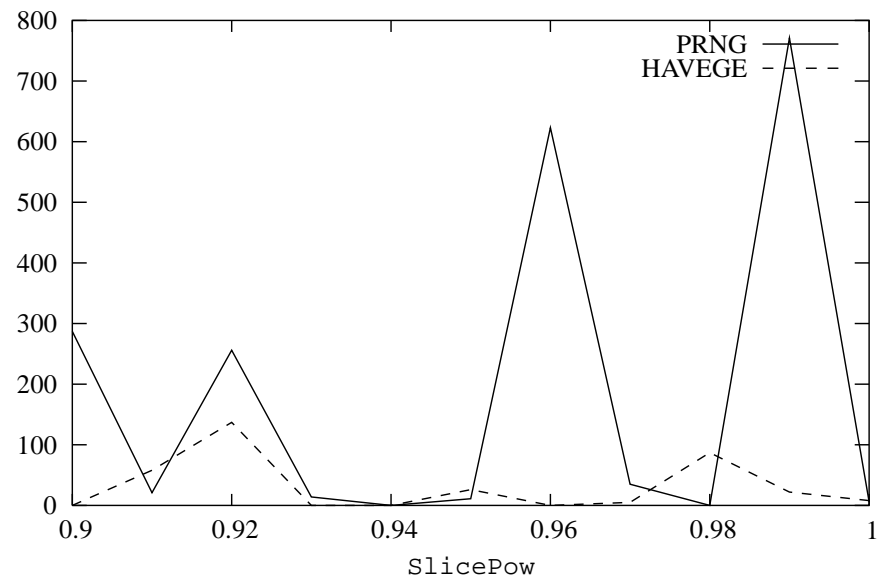


Figure 3: Plot of the origination time of the organism with maximal complexity, as a function of parsimony pressure. PRNG = pseudo random number generator, and HAVEGE is the entropy harvester mentioned in the text.

Standish, R. K. (2003). Open-ended artificial evolution. *International Journal of Computational Intelligence and Applications*, 3:167.